SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.035 Computer Vision

Fake Image Detection

1D Project Report

Team Members
Wang Han 1004520
Yap Zhi Han 1004570
Yu Peijia 1004538
Yang Zhonghao 1004516

1.Introduction

Image editing is the process of modifying images using the software. With the development in image editing technology, digitally modified/created images are hard to distinguish from original images. Although it is widely adopted in various fields such as film, magazine, and blog posts, these images could be misused.

The project aims to design a computer vision model that can distinguish between digitally modified and unmodified images, paying particular attention to real images captured by cameras. Two types of digital image modifications are studied, which are feature duplication and movement, and processing images with software like Meitu.

Fake image detection involves steps of image preprocessing, training stage, and post-processing. All three steps are closely related to the concepts in computer vision. For image preprocessing, we can do image augmentation or transformation to enhance the stability of our model. To improve the accuracy, batch normalization tools can also be used as it standardizes the inputs to a layer for each mini-batch. For the training stage, CNN models such as AlexNet, VGG, and ResNet can be used. For the activation function, we will use Relu for fast convergence. And choose max pooling for the pooling layer for extracting sharp features. For the post-processing stage, to avoid overfitting we can do random dropout of nodes, regularization of the loss function, or early stopping of the training stage.

2. Dataset and Data Preprocessing

2.1 Dataset

2.1.1 CASIA v2.0 - feature duplication and movement

CASIA v2.0 is the dataset for forgery image classification, which contains 7491 authentic images and 5123 tampered images. It is generated by a group of researchers from the Institute of Automation, Chinese Academy of Sciences. Images in the authentic set were mostly collected from the Corel image dataset and others were taken by the researchers' own cameras. The authentic set can be roughly clustered into 8 categories according to image content (scene, animal, architecture, character, plant, article, nature and texture). The tampered images are generated only by using crop-and-paste operation under Adobe Photoshop on these authentic images, so these images can also be called as spliced images.
The dataset used for our project is generated from CASIA v2.0. To balance the influences of authentic and tampered images on model training and evaluation, an algorithm is designed and implemented in `choose_pic.py`*(The code can be viewed at code/choose_pic.py)* to randomly choose 5000 authentic images and 5000 tampered images. An 8: 2 train test split ratio is applied to the dataset, and 10% of the data in the training dataset is used as the validation data. There are 7200 images in the training dataset, 800 images in the validation dataset, and 2000 images in the testing dataset. Besides, in the rest 2491 authentic images and

123 tampered images, 100 authentic and 100 tampered images are chosen to generate a small testing dataset for further evaluation, calculating precision, recall, and f1 score.

2.1.2 Meitu dataset - image edited by software like Meitu

Another general image modification method is applying image editing software like Meitu to images. However, no widely used dataset is found. But a group of researchers in the Department of Computer Science and Department of Operations Research & Information Engineering, Cornell Tech is also studying how to build a CNN model to detect human-edited images. They collect 1081 authentic human faces from different datasets like Real and Fake Face Detection Dataset and Flickr-Faces-HQ Dataset, and apply editings like smoothing, teeth whitening, and feature swapping manually using various image editing software(*Table 2.1.2.1*). The dataset contains 1081 authentic images and 960 human-edited images.

Source	Editing software	Edits performed
Flickr-Faces-HQ Dataset	Adobe	Smoothing
Facebook	Facetune	Smoothing, Teeth whitening
Helen Dataset	Pixelmator	Warping, Bumping, Pinching
Real and Fake Face Detection Dataset		Feature swapping

Table 2.1.2.1

In our project, 960 authentic and 960 human-edited images are randomly chosen. An 8: 2 train test split ratio is applied to the dataset, and 10% of the data in the training dataset is used as the validation data.

2.1.3 Small dataset for data preprocessing method comparison and further exploration

To compare whether the best model which works on the feature duplication dataset also performs well on the Meitu dataset, 960 authentic and 960 tampered images are randomly chosen using the algorithm in `choose_pic.py`, to match the size of the Meitu dataset. Similarly, An 8: 2 train test split ratio is applied to the dataset, and 10% of the data in the training dataset is used as the validation data. For convenience, this small dataset is also used for data preprocessing method evaluation.

2.2 Data preprocessing

Three image transformation techniques are considered, including local binary patterns(LBP), error level analysis(ELA) and discrete wavelet transform(DWT). In order to see how each image processing method influences the model accuracy on feature duplication and movement images, we train VGG10 on images from the dataset mentioned in 2.1.3 preprocessed by each method mentioned above and choose the method which leads to the best model performance and apply it to the images for all models.

2.2.1 No Image Transformation Techniques

The image is directly converted to a tensor without any image transformation techniques. A batch size of 64 is applied due to the GPU limitation. The VGG 10 is trained for 30 epochs, and the lowest validation loss happens in epoch 5, with a test accuracy of 64.32%. *(The code can be viewed at code/Image Transformation/VGG10-copymove-notransformation.ipynb)*

2.2.2 Local Binary Pattern

The Local Binary Pattern is applied to the image and then the image is converted to a tensor. LBP is computationally simple but can only be applied to grayscale images. LBP usually considers a 3*3 block around every pixel and sets a threshold to label 8 pixels around the central one. LBP code for each pixel is the weighted sum of 8 labels around it.

The model VGG10 is slightly modified because the image processed by LBP only has one channel, so the first layer of the model should be the convolutional layer with the input channel as 1.

A batch size of 64 is applied due to the GPU limitation. The VGG 10 is trained for 30 epochs, and the lowest validation loss happens in epoch 7, with a test accuracy of 43.49%. *(The code can be viewed at code/Image Transformation/VGG10-copymove-lbp.ipynb)*

2.2.3 Error Level Analysis

Error Level Analysis calculates the differences between the original image and the image compressed at a known error level. The attribute of ELA is that it is uniform across the authentic image, but for a tampered image, this difference would be higher for the areas of the image that someone has modified.

A batch size of 64 is applied but no GPU limitation is detected under this circumstance. The VGG 10 is trained for 30 epochs, and the lowest validation loss happens in epoch 8, with a test accuracy of 87.24%. *(The code can be viewed at code/Image Transformation/VGG10-copymove-ela.ipynb)*

2.2.4 Discrete Wavelet Transform

DWT is used to capture location information concerning time as well as frequency information, which represents the edges of an image in different dimensions. Three coefficients which represent horizontal detail, vertical detail, and diagonal detail are extracted and concatenated to three channels.

A batch size of 64 is applied but no GPU limitation is detected under this circumstance. The VGG 10 is trained for 30 epochs, and the lowest validation loss happens in epoch 11, with a test accuracy of 67.45%. *(The code can be viewed at code/Image Transformation/VGG10-copymove-dwt.ipynb)*

	No techniques	LBP	ELA	DWT
GPU occupied	50G	50G	14G	45G
Best epoch	5	7	8	11
Test accuracy	64.32%	43.49%	87.24%	67.45%

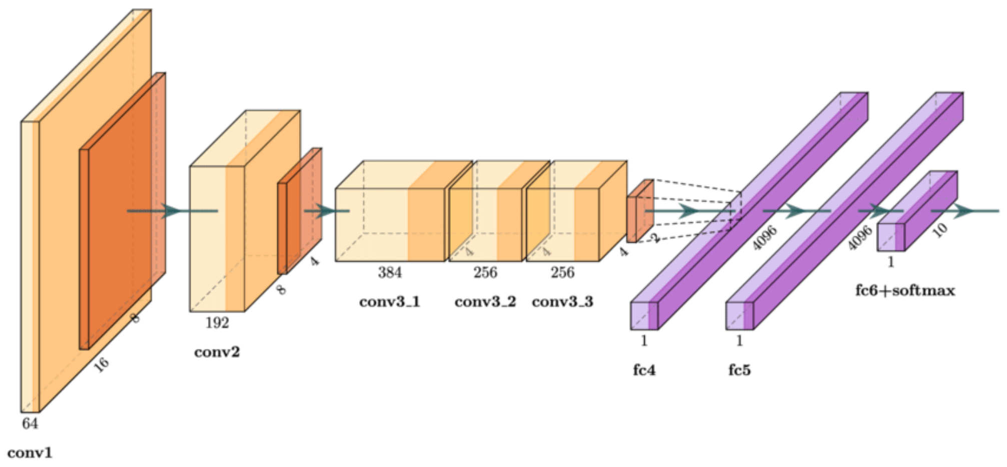
Table 2.2.4.1 Summary of evaluation results for image transformation techniques

Based on the evaluation result, we can find that the image processed with LBP reaches the lowest accuracy. This may be because some feature in the image is lost when converting the RGB image to a grayscale image. ELA performs better than no image transformation techniques and DWT with a much higher test accuracy. Besides, the model trained with ELA requires much less GPU storage given the same data loader size and batch size. As a result, ELA is chosen for image transformation for all the models we want to study.

3. Model Architecture and Performance

3.1 Alexnet

3.1.1 AlexNet Model Architecture



The AlexNet model consists of 5 convolutional layers and 3 fully connected layers, each convolutional layer consists of basic 2D convolution, max pooling, 2D batch normalization and the activation function. The original AlexNet model implements ReLU as the activation, to avoid the "dying ReLU" issue, we used Leaky ReLU function instead. *(The code can be viewed at code/AlexNet/alextnet.ipynb)*

3.1.2 Performance under Different Learning Rates

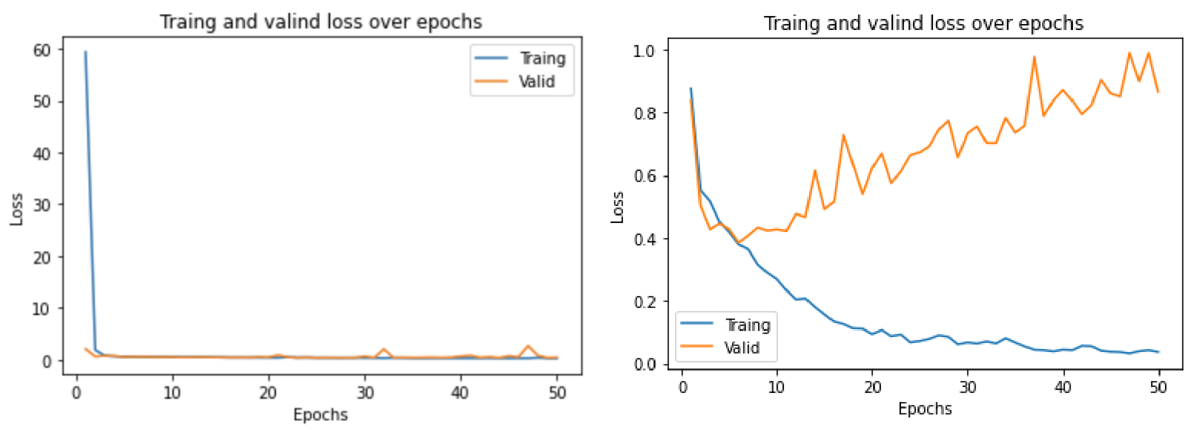


Figure 3.1.2.1: Training and validation loss of AlexNet under learning rate = 1e-2 and a zoom in of validation loss

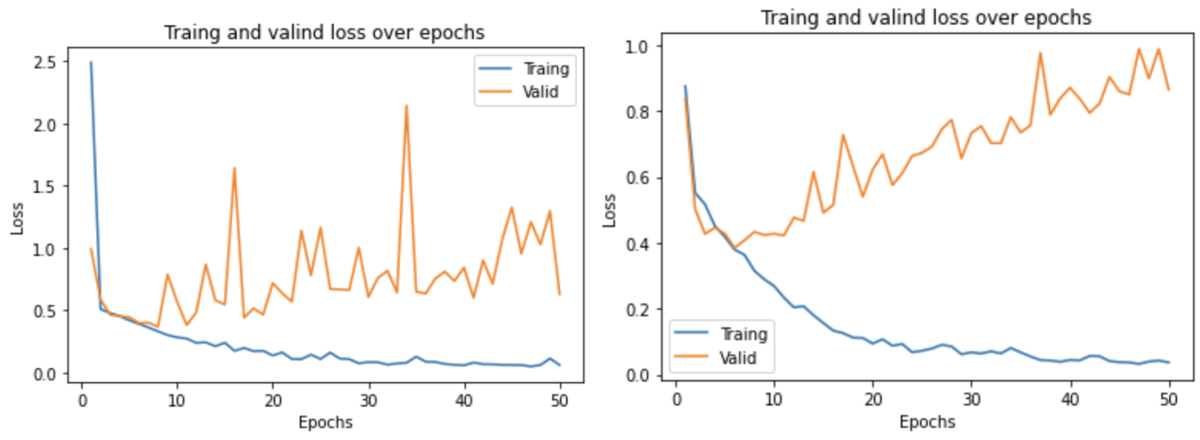


Figure 3.1.2.2: Training and validation loss of AlexNet under learning rate = 1e-3 and 1e-4

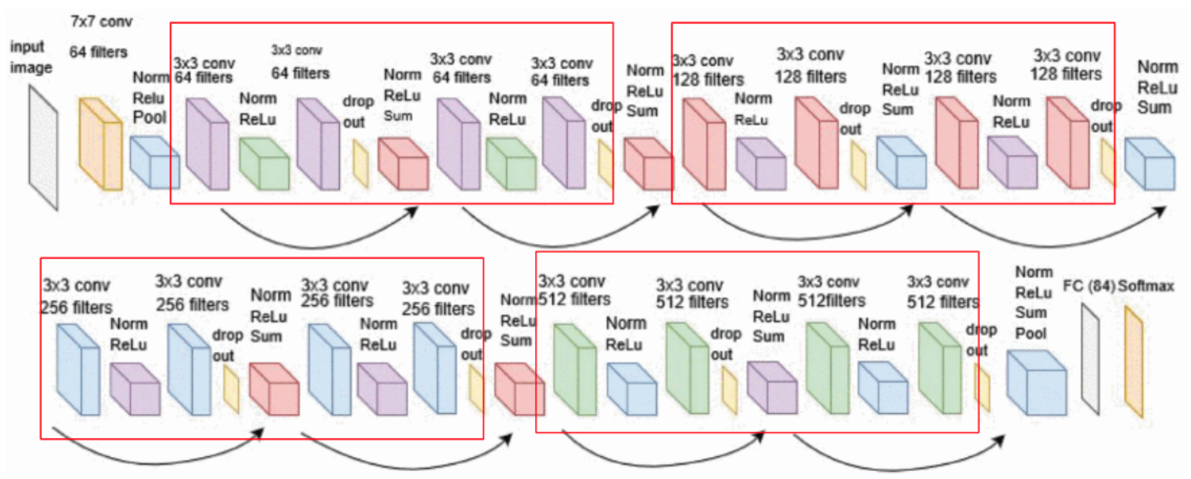
Figure 3.1.2.1 and Figure 3.1.2.2 illustrate the training and validation loss of the AlexModel under different learning rates. We performed three trials on each learning rate and took the average loss values as follows:

Learning rate = 1e-2 | Average test loss = 0.341 | Average test accuracy = 85.85%
Learning rate = 1e-3 | Average test loss = 0.364 | Average test accuracy = 85.83%
Learning rate = 1e-4 | Average test loss = 0.413 | Average test accuracy = 83.82%

Based on our observation, AlexNet achieves high performance with LR = 1e-2 and 1e-3.

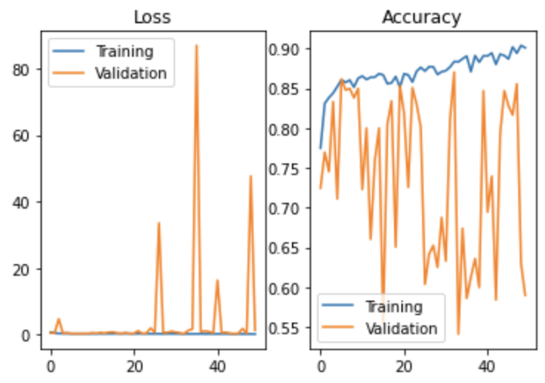
3.2 Resnet

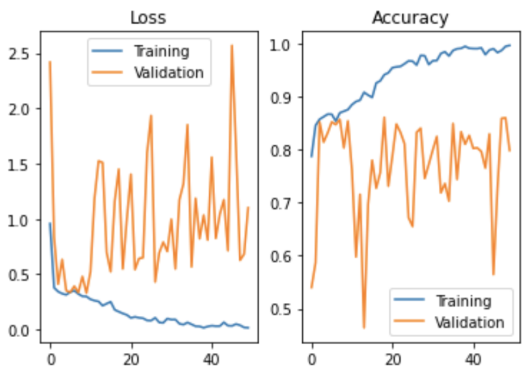
3.2.1 ResNet Model Architecture



Among all the versions of ResNet, we chose to use ResNet-18 because the larger models have unreasonably high computation times and insufficient memory on our machines to use them. There are a total of 4 blocks in the ResNet-18 model, each containing 4 convolution layers, increasing in depth per block from 64 to 512. It uses ReLU as the activation function, and batch normalization is applied after each convolution layer. In addition, skip connection is applied in ResNet to alleviate gradient vanishing issues. *(The code can be viewed at code/ResNet/resnet.ipynb)*

3.2.2 Performance of ResNet under Different Learning Rates





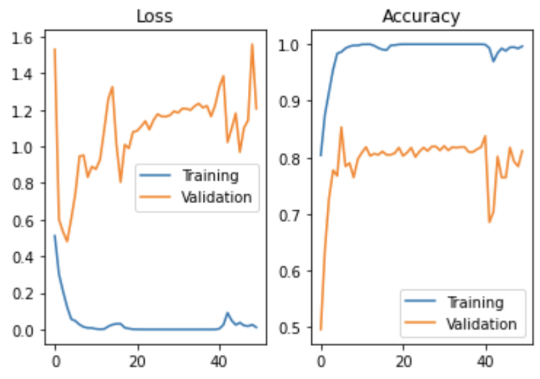


Figure 3.2.2.1: Training and validation loss of ResNet-18 under learning rate = 1e-2, 1e-3 and 1e-4

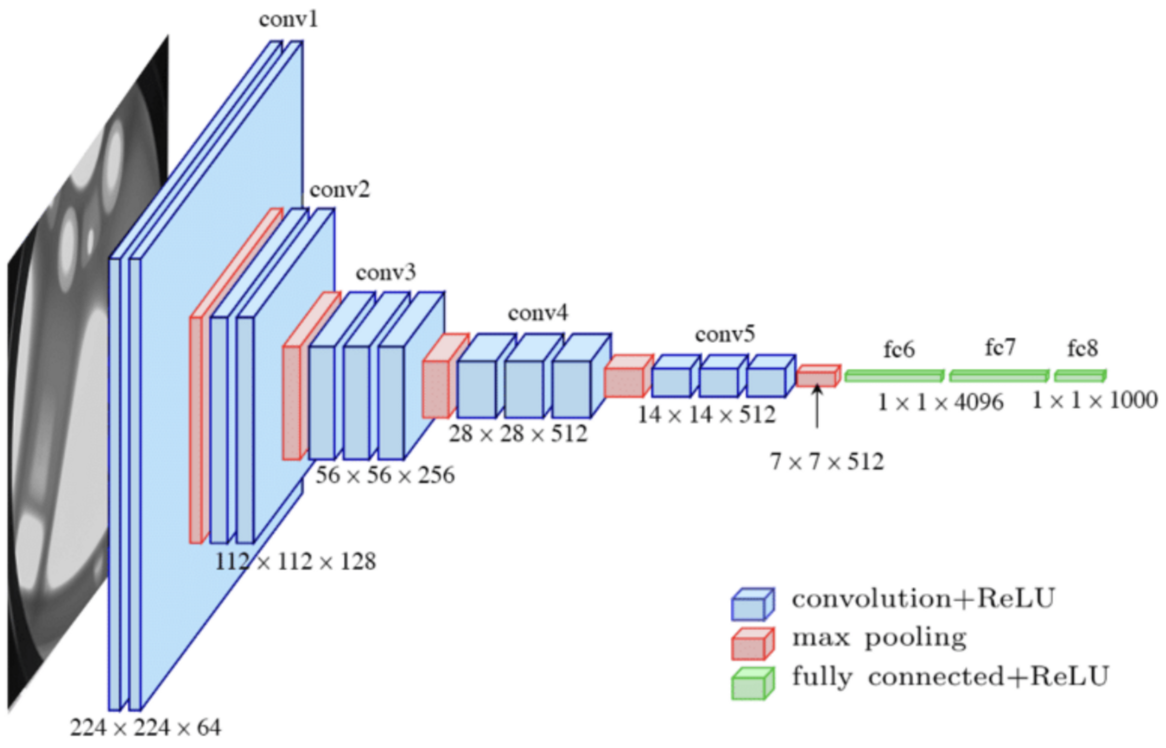
We tested ResNet18 on different learning rates, the training and validation loss were calculated as the average loss of three trials as follows:

Learning rate = 1e-2 | Average test loss = 0.284 | Average test accuracy = 88.21%
Learning rate = 1e-3 | Average test loss = 0.300 | Average test accuracy = 86.36%
Learning rate = 1e-4 | Average test loss = 0.363 | Average test accuracy = 83.83%

We found that the best results were with a learning rate of 1e-2. As expected, with a more complex network architecture as compared to AlexNet, ResNet performs better with higher test accuracy.

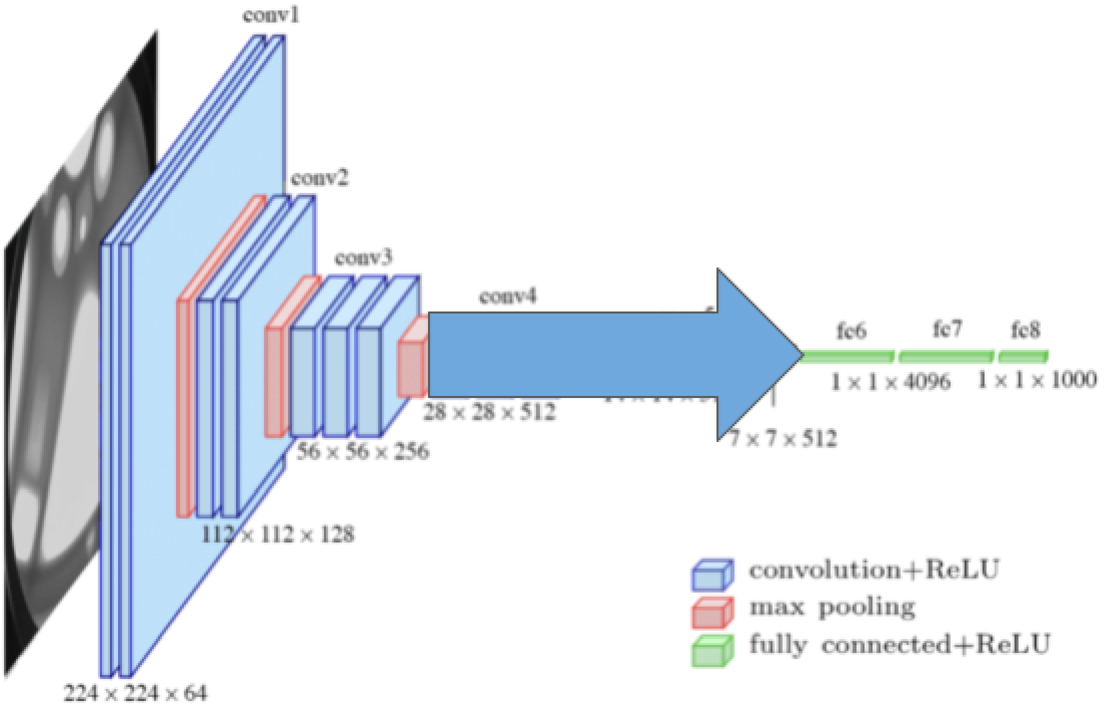
3.3 VGG

3.3.1 VGG-16 Model Architecture



The overall VGG-16 model consists of 13 convolution layers(grouped into 5 clusters) and 3 fully connected layers. That is why it is named as VGG-16, because it has 16 trainable layers. Because it has a relatively deep structure, its overfitting issue also becomes serious. Our solution to this problem is to eliminate the last two clusters of convolution layers to get a VGG model with 10 trainable layers. We call this new model VGG-10. *(The code can be viewed at code/VGG16/VGG16.ipynb)*

3.3.2 VGG-10 Model Architecture



The overall VGG-10 model consists of 7 convolution layers (grouped into 3 clusters) and 3 fully connected layers. Which has 6 less trainable layers than VGG-16. To compare these two models' performance, we trained both models for 50 epochs using the pretrained parameters and a learning rate of 1e-3 (the learning rate that achieves the best performance for both two models). *(The code can be viewed at code/VGG10/VGG10.ipynb)*

3.3.3 Performance Comparison in LR = 1e-3

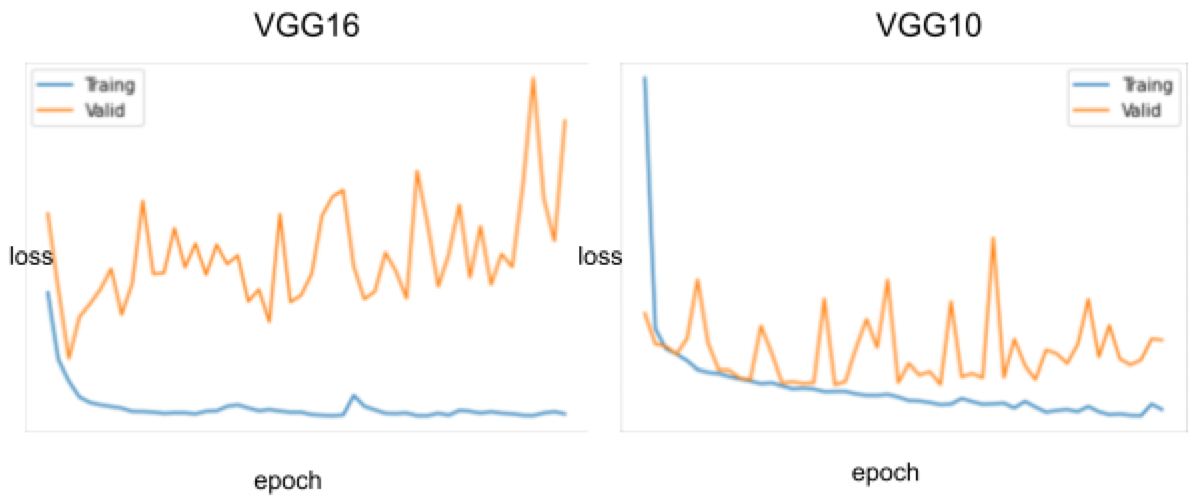
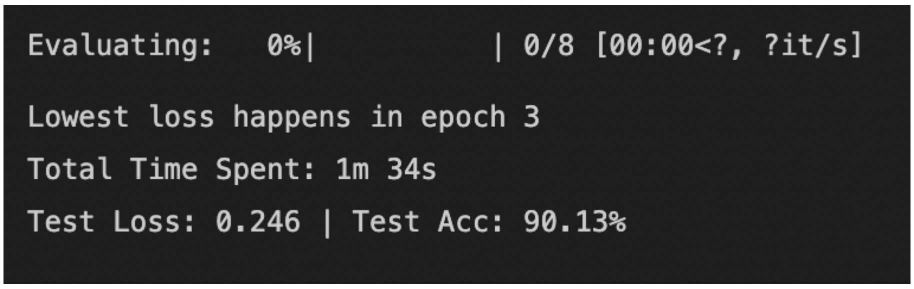


Figure 3.3.3.1

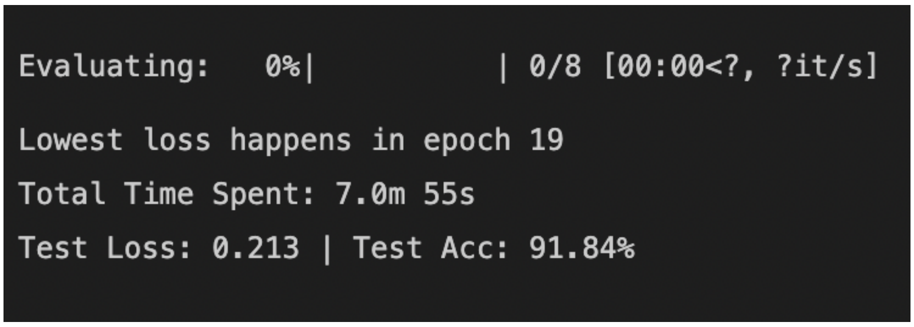
Figure 3.3.3.1 displays the training and validation loss over 50 epoches for VGG16 and VGG10. As we can see the validation loss for vgg16 model increases very soon after training for a few epochs. However, this problem is much alleviated for the vgg10 model.



```
Evaluating:   0%|          | 0/8 [00:00<?, ?it/s]

Lowest loss happens in epoch 3
Total Time Spent: 1m 34s
Test Loss: 0.246 | Test Acc: 90.13%
```

Figure 3.3.3.2 Test accuracy for VGG-16



```
Evaluating:   0%|          | 0/8 [00:00<?, ?it/s]

Lowest loss happens in epoch 19
Total Time Spent: 7.0m 55s
Test Loss: 0.213 | Test Acc: 91.84%
```

Figure 3.3.3.3 Test accuracy for VGG-10

The advantages of the VGG-10 model could also be revealed in test accuracy. The VGG-16 model achieved its best performance in epoch3, after Epoch 3, it began to overfit. In the end, it got a test accuracy of 90.13%. However, the VGG-10 only got its best performance in Epoch 19. After training for more episodes, it also has a higher test accuracy of 91.84%.

The only disadvantage of VGG-10 is that it takes longer than the VGG-16. This may be because it has a simpler model, so it needs to try harder to fit the data.

A further evaluation between VGG-10 and VGG-16 has been done using a smaller dataset with 100 authentic and 100 tampered images mentioned in *Section 2.1.1*. The evaluation result is listed in *Table 3.3.3.4*. Here the situation is worse when fake images are not detected, so the label fake is positive, and the label real is negative. From the result, we can see that VGG-10 performs better than VGG-16 especially in detecting fake images. *(The code can be viewed at code/further_evaluation.ipynb)*

	VGG-10	VGG-16
correctly predicted real images	85	84
correctly predicted fake images	96	93
f1 score	0.91	0.89
recall	0.96	0.93
precision	0.86	0.85

Table 3.3.3.4

3.4 Conclusion on Model Performance

Alexnet only has 8 trainable layers, which means it could not compare to the performance of the other two models. It uses the largest learning rate (1e-2 compared to 1e-3 in another two models) to achieve its best performance, but just see from the previous *Section 3.1.2*, we could find that the initial training loss for AlexNet with 1e-2 is extremely high, was about 60. This is quite unstable and not what we want.

ResNet-18 has deeper models which require more data to reduce overfitting. But because of our resource limit, we could not provide and run on large datasets, the problem of overfitting in resnet is still serious.

Finally we choose VGG to use in the further exploration. For our task, VGG-10 is just the suitable model, not too simple, could not learn the data very well, not too complex to have a tendency to overfit.

4. Further Exploration

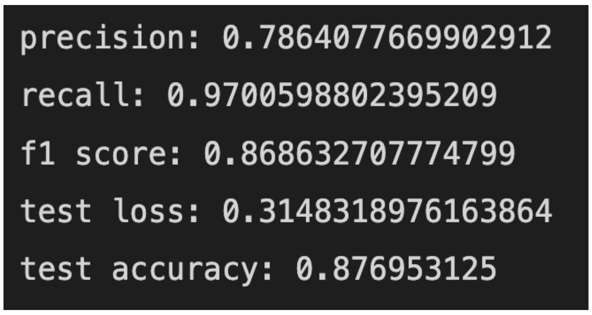
4.1 Performance in Meitu Dataset

4.1.1 Apply the Model to Meitu Dataset

After selecting VGG-10 to be the best model in copy-and-move fake image detection, we want to figure out whether the model can be applied to images that are modified with tools like Meitu also.

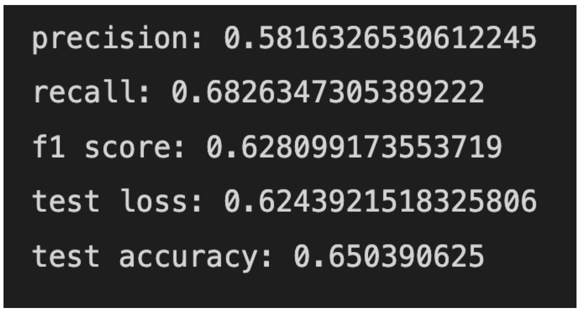
The dataset mentioned in *Section 2.1.2 and Section 2.1.3* are used for comparison.

4.1.2 Comparison of Performance



```
precision: 0.7864077669902912
recall: 0.9700598802395209
f1 score: 0.868632707774799
test loss: 0.3148318976163864
test accuracy: 0.876953125
```

Figure 4.1.2.1 CASIA v2.0



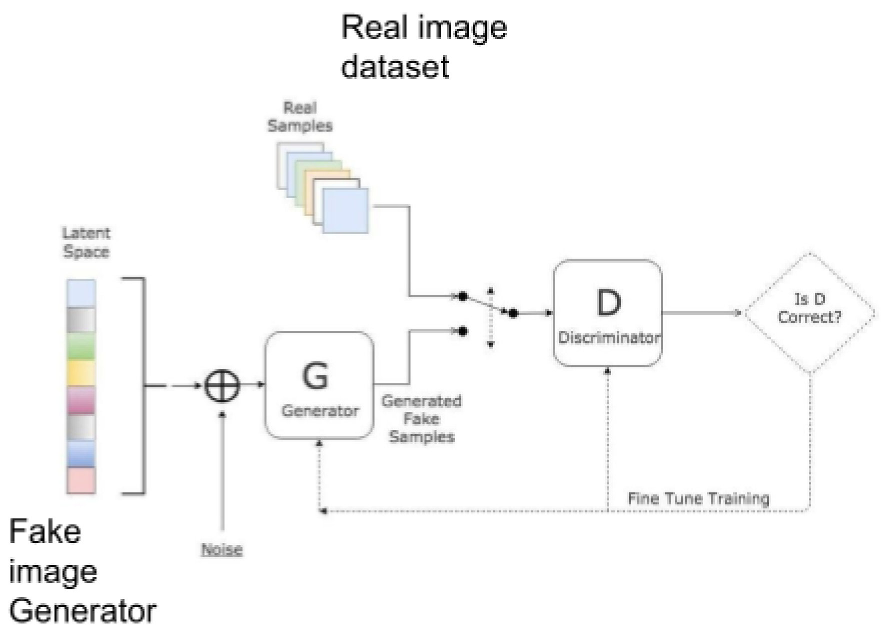
```
precision: 0.5816326530612245
recall: 0.6826347305389222
f1 score: 0.628099173553719
test loss: 0.6243921518325806
test accuracy: 0.650390625
```

Figure 4.1.2.2 Meitu

After training separately on different dataset, we found that even using a smaller dataset, the VGG-10 model could still achieve an accuracy of 87.70% in copy-and-move fake image detection. However, the accuracy drops a lot in the Meitu dataset which was only 65.04%. *(The code can be viewed at the folder code/CASIA&Meitu Dataset Comparison/)*

4.2 Possible Reasons and Solutions

The dataset of images processed by tools like Meitu is relatively small, and we may utilize models like GAN to generate these fake images to train our classifier.



The changes of fake images in this dataset may be smaller compared with fake images from CASIA 2.0, so ELA may not be the best data pre-processing method. Other image transformation methods can be used for image preprocessing, like liquify.

The structure of VGG-10 may not be able to detect slight changes in a picture, we may try other deeper models like Resnet, whose performance was poorer in copy-and-model fake image detection, but may achieve a better performance in Meitu dataset.

5. References

Chang, X., Wu, J., Yang, T., & Feng, G. (2020, July). DeepFake face image detection based on improved VGG convolutional neural network. In *2020 39th Chinese Control Conference (CCC)* (pp. 7252-7256). IEEE.https://ieeexplore.ieee.org/abstract/document/9189596?casa_token=Yrbu9vWKcQ4AAAAA:C8fO1eO1IcO21uZVvsaaP5voXhMxtleGxNdAGxVTUSrnQO5dPOKc4ffBDKsTXUMTS7KO5QF0HnV0lw

Mo, H., Chen, B., & Luo, W. (2018, June). Fake faces identification via convolutional neural network. In *Proceedings of the 6th ACM workshop on information hiding and multimedia security* (pp. 43-47).https://dl.acm.org/doi/abs/10.1145/3206004.3206009?casa_token=RiHiCiR7qSEAAAAA:UwOTTWPBfZX4Oe8jlbwGDW4auhbPAkxiO_4HSy5fNE8Dp5kF012xCMW2Zmx4A2TXyDErvKNjL2yxCA

Tariq, S., Lee, S., Kim, H., Shin, Y., & Woo, S. S. (2018, January). Detecting both machine and human created fake face images in the wild. In *Proceedings of the 2nd international workshop on multimedia privacy and security* (pp. 81-87) https://dl.acm.org/doi/abs/10.1145/3267357.3267367?casa_token=aFT4Q8rfCr4AAAAA:2p8dr0cHfC7DxE21QJMvTj0afiq_dGMuYJbc0qbdmrEhVMEgWt80RVrcousoK-u3261dMJGpqPjr5A

Ghai, A., Kumar, P., & Gupta, S. (2021). A deep-learning-based image forgery detection framework for controlling the spread of misinformation. *Information Technology & People*. https://doi.org/10.1108/itp-10-2020-0699